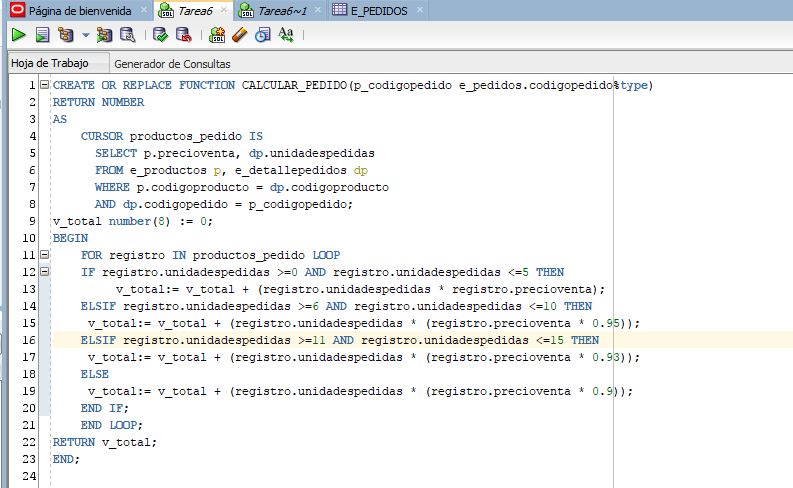
## **Tarea para BD06.**

#### **Actividad 1.**

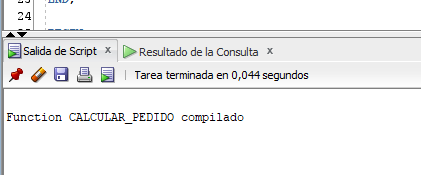
Realizaremos una **función** llamada **CALCULAR\_PEDIDO** a la que le daremos un código de pedido como entrada y nos devuelva como salida el total de ese pedido. A cada producto del pedido  se le realizará un descuento sobre el precio de venta que dependerá del nº de unidades pedidas:

* Si el nº de unidades está entre 0 y 5: no habrá descuento
* Si en nº de unidades pedidas está entre 6 y 10 el descuento es del 5%
* Si el nº de unidades pedidas está entre 11 y 15 el descuento es del 7%
* Si el nº de unidades pedidas es >15 el descuento será del 10%.

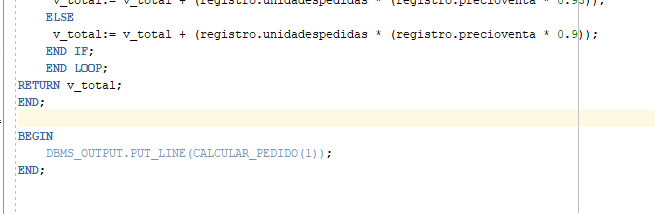
El total de un producto será el resultado de multiplicar el precio de venta (con el descuento realizado) por en nº de unidades pedidas.



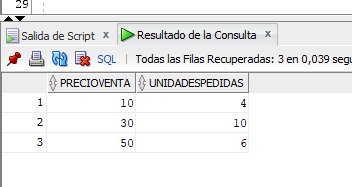
Al ejecutar la función nos dice que se compila correctamente:



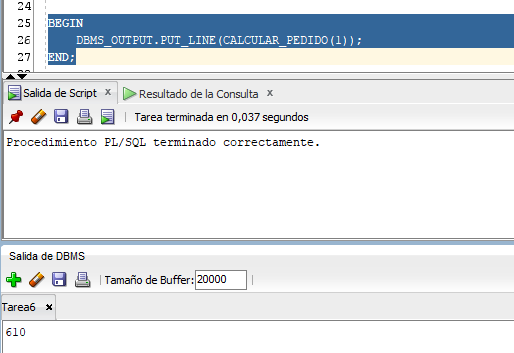
Para añadir el código del pedido como entrada añado un DBMS\_OUTPUT y le paso como parámetro un 1.



El pedido 1 se corresponde con la siguiente tabla



Ejecución DBMS\_OUTPUT:



## COMPROBACIÓN:

El resultado es 610, la suma de:

* 10 \*4 = 40 No hay descuento
* (30 \* 0,95) \* 10 = 285
* (50 \* 0,95) \* 6 = 285

# Sentencia:

CREATE OR REPLACE FUNCTION CALCULAR\_PEDIDO(p\_codigopedido e\_pedidos.codigopedido%type)

RETURN NUMBER

AS

CURSOR productos\_pedido IS

SELECT p.precioventa, dp.unidadespedidas

FROM e\_productos p, e\_detallepedidos dp

WHERE p.codigoproducto = dp.codigoproducto

AND dp.codigopedido = p\_codigopedido;

v\_total number(8) := 0;

BEGIN

FOR registro IN productos\_pedido LOOP

IF registro.unidadespedidas >=0 AND registro.unidadespedidas <=5 THEN

v\_total:= v\_total + (registro.unidadespedidas \* registro.precioventa);

ELSIF registro.unidadespedidas >=6 AND registro.unidadespedidas <=10 THEN

v\_total:= v\_total + (registro.unidadespedidas \* (registro.precioventa \* 0.95));

ELSIF registro.unidadespedidas >=11 AND registro.unidadespedidas <=15 THEN

v\_total:= v\_total + (registro.unidadespedidas \* (registro.precioventa \* 0.93));

ELSE

v\_total:= v\_total + (registro.unidadespedidas \* (registro.precioventa \* 0.9));

END IF;

END LOOP;

RETURN v\_total;

END;

BEGIN

DBMS\_OUTPUT.PUT\_LINE(CALCULAR\_PEDIDO(1));

END;

#### **Actividad 2**

Realizaremos un **procedimiento** llamado **CALCULAR\_CLIENTE**que dándole como entrada un código de cliente y un año, nos devuelva como salida dos parámetros : el total facturado por los pedidos que haya pagado y el total facturado por los pedidos que no haya pagado durante ese año ese cliente. Sólo se tendrán en cuenta los pedidos que se hayan entregado ya.

Sólo tendremos en cuenta los clientes que hayan realizado más de dos pedido en ese año.

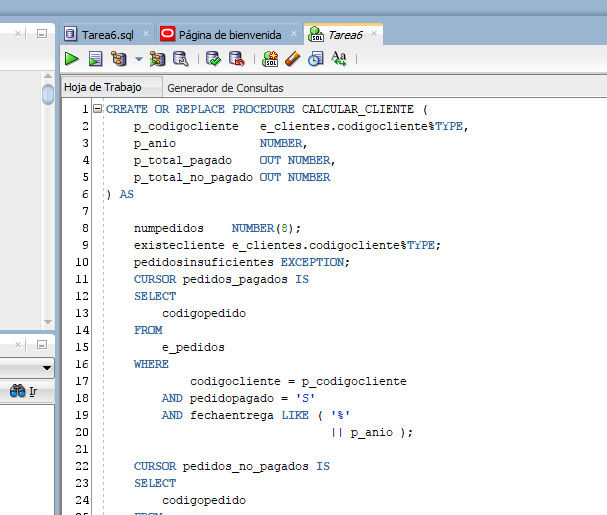
Si el cliente no existe saltaremos una excepción, visualizaremos un mensaje de que ese cliente no existe y finalizará el procedimiento devolviendo como salida -1 en ambos totales.

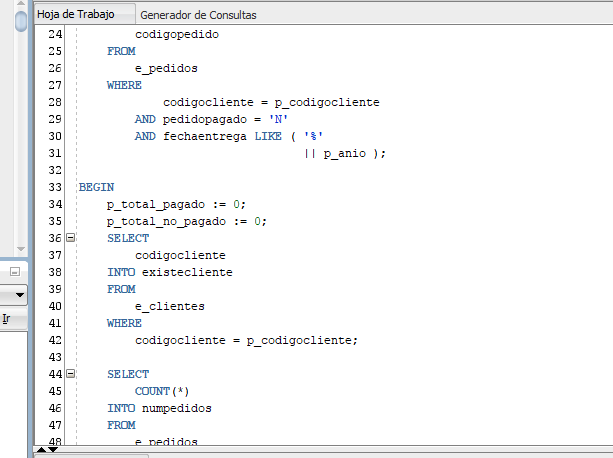
Si el cliente no tiene pedidos durante ese año, saltaremos una excepción y visualizaremos un mensaje de que ese cliente tiene no tiene pedidos ese año y finalizará el procedimiento devolviendo como salida -1 en ambos totales

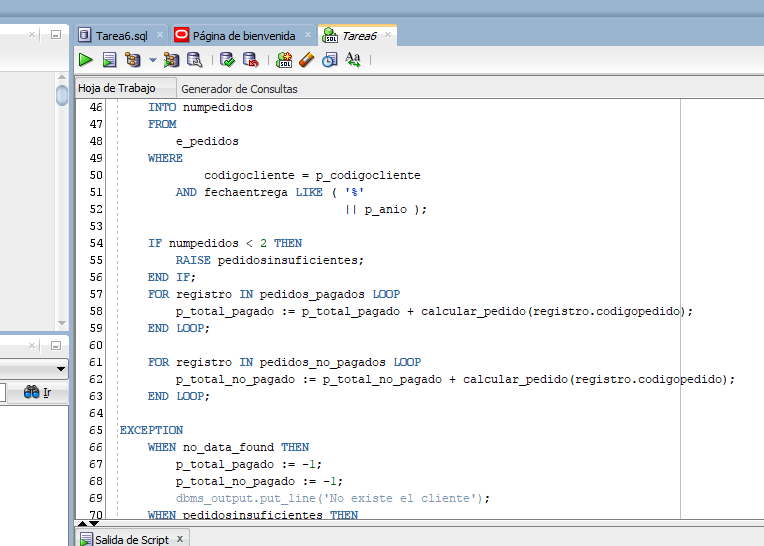
Por cada pedido realizaremos una llamada a la función CALCULAR\_PEDIDO, realizada en la actividad 1,  que nos calculará el total de cada pedido. Estos totales se irán acumulando en el total de los pedidos pagados o en el total de los pedidos no pagados.

Finalmente devolveremos como parámetros dichos totales.

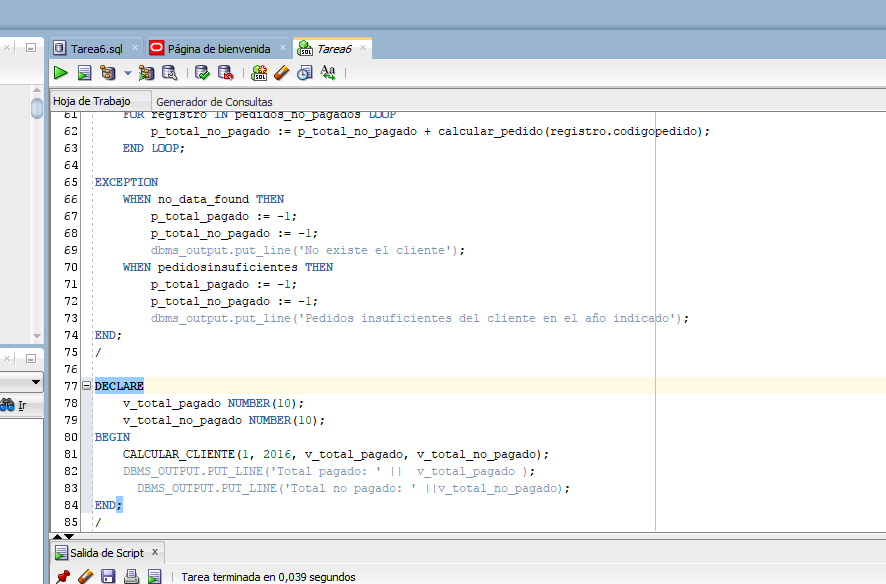
## Capturas de pantalla







Fin del procedimiento y añado un nuevo bloque para pasar los parámetros de entrada del código y el año y que devuelva el total facturado por los pedidos que haya pagado(v\_total\_pagado) y el total facturado por los pedidos que no haya pagado(v\_total\_no\_pagado) durante ese año ese cliente.



# Sentencia:

CREATE OR REPLACE PROCEDURE CALCULAR\_CLIENTE (

p\_codigocliente e\_clientes.codigocliente%TYPE,

p\_anio NUMBER,

p\_total\_pagado OUT NUMBER,

p\_total\_no\_pagado OUT NUMBER

) AS

numpedidos NUMBER(8);

existecliente e\_clientes.codigocliente%TYPE;

pedidosinsuficientes EXCEPTION;

CURSOR pedidos\_pagados IS

SELECT

codigopedido

FROM

e\_pedidos

WHERE

codigocliente = p\_codigocliente

AND pedidopagado = 'S'

AND fechaentrega LIKE ( '%'

|| p\_anio );

CURSOR pedidos\_no\_pagados IS

SELECT

codigopedido

FROM

e\_pedidos

WHERE

codigocliente = p\_codigocliente

AND pedidopagado = 'N'

AND fechaentrega LIKE ( '%'

|| p\_anio );

BEGIN

p\_total\_pagado := 0;

p\_total\_no\_pagado := 0;

SELECT

codigocliente

INTO existecliente

FROM

e\_clientes

WHERE

codigocliente = p\_codigocliente;

SELECT

COUNT(\*)

INTO numpedidos

FROM

e\_pedidos

WHERE

codigocliente = p\_codigocliente

AND fechaentrega LIKE ( '%'

|| p\_anio );

IF numpedidos < 2 THEN

RAISE pedidosinsuficientes;

END IF;

FOR registro IN pedidos\_pagados LOOP

p\_total\_pagado := p\_total\_pagado + calcular\_pedido(registro.codigopedido);

END LOOP;

FOR registro IN pedidos\_no\_pagados LOOP

p\_total\_no\_pagado := p\_total\_no\_pagado + calcular\_pedido(registro.codigopedido);

END LOOP;

EXCEPTION

WHEN no\_data\_found THEN

p\_total\_pagado := -1;

p\_total\_no\_pagado := -1;

dbms\_output.put\_line('No existe el cliente');

WHEN pedidosinsuficientes THEN

p\_total\_pagado := -1;

p\_total\_no\_pagado := -1;

dbms\_output.put\_line('Pedidos insuficientes del cliente en el año indicado');

END;

/

DECLARE

v\_total\_pagado NUMBER(10);

v\_total\_no\_pagado NUMBER(10);

BEGIN

CALCULAR\_CLIENTE(1, 2015, v\_total\_pagado, v\_total\_no\_pagado);

DBMS\_OUTPUT.PUT\_LINE('Total pagado: ' || v\_total\_pagado );

DBMS\_OUTPUT.PUT\_LINE('Total no pagado: ' ||v\_total\_no\_pagado);

END;

/

#### **Actividad 3.**

Queremos crear los siguientes disparadores:

3.1.   Un disparador llamado **DISP\_PEDIDOS** que salte al insertar o actualizar  en la tabla **pedidos**para que no nos deje insertar o actualiza si se produce uno de los siguientes casos:

* + - La fecha de pedido debe ser menor que la fecha esperada de entrega y la fecha de entrega.
    - El estado de un pedido sólo puede tener los valores P, E o D (Pendiente de entregar, Entregado o Devuelto)
    - La columna PedidoPagado sólo puede tener los valores : S o N (Pagado Si, Pagado No)

Se lanzará una excepción mediante la cual el registro no se inserte y visualiza el mensaje adecuado

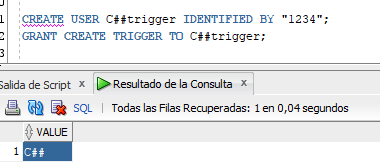
Como la tabla es propiedad de SYS, tengo que crear un nuevo usuario ya que da error al ejecutar disparadores ya que el usuario SYS es el usuario del sistema y se usa para realizar tareas críticas del sistema.

Creo un usuario con el prefijo del sistema

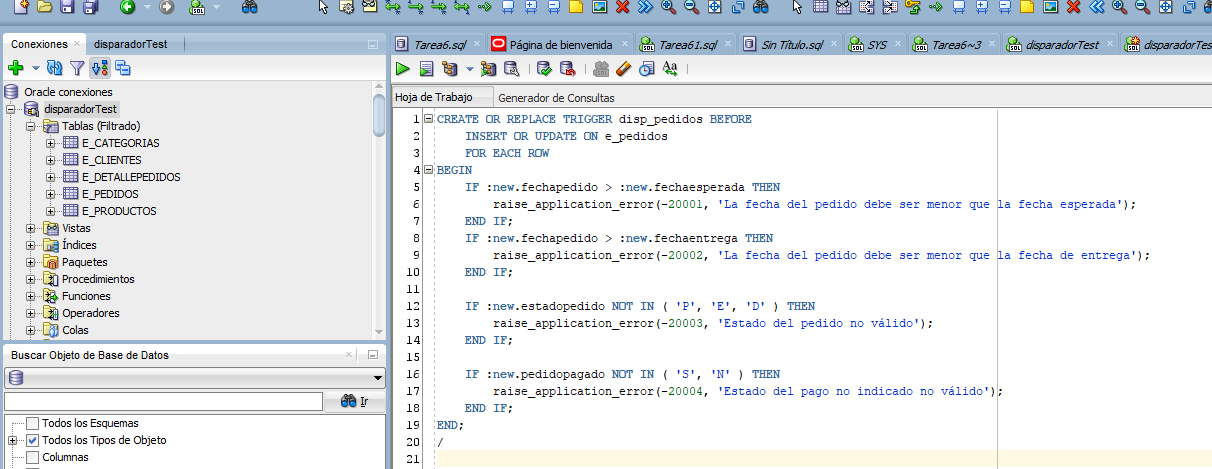
SELECT value FROM v$parameter WHERE name = 'common\_user\_prefix';

CREATE USER C##trigger IDENTIFIED BY "1234";

GRANT CREATE TRIGGER TO C##trigger;



Creo el usuario con todas las tablas, función y procedimiento para poder ejecutar el disparador… y lo testeo insertando valores en la tabla e\_pedidos con un insert.



# Sentencia:

CREATE OR REPLACE TRIGGER disp\_pedidos BEFORE

INSERT OR UPDATE ON e\_pedidos

FOR EACH ROW

BEGIN

IF :new.fechapedido > :new.fechaesperada THEN

raise\_application\_error(-20001, 'La fecha del pedido debe ser menor que la fecha esperada');

END IF;

IF :new.fechapedido > :new.fechaentrega THEN

raise\_application\_error(-20002, 'La fecha del pedido debe ser menor que la fecha de entrega');

END IF;

IF :new.estadopedido NOT IN ( 'P', 'E', 'D' ) THEN

raise\_application\_error(-20003, 'Estado del pedido no válido');

END IF;

IF :new.pedidopagado NOT IN ( 'S', 'N' ) THEN

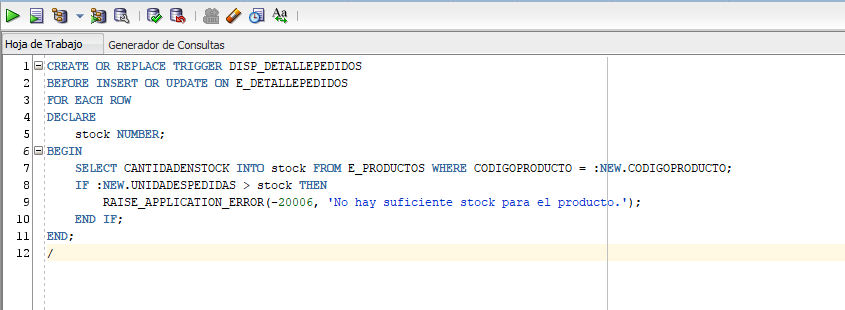
raise\_application\_error(-20004, 'Estado del pago no indicado no válido');

END IF;

END;

/

3.2. Un disparador llamado **DISP\_DETALLEPEDIDOS**, en el que al insertar o actualizar en la tabla **detallepedidos** controle que la cantidad pedida del producto sea menor que la cantidad en stock de dicho producto. Se lanzará una excepción mediante la cual el registro no se inserte y visualiza el mensaje adecuado.



# Sentencia:

CREATE OR REPLACE TRIGGER DISP\_DETALLEPEDIDOS

BEFORE INSERT OR UPDATE ON E\_DETALLEPEDIDOS

FOR EACH ROW

DECLARE

stock NUMBER;

BEGIN

SELECT CANTIDADENSTOCK INTO stock FROM E\_PRODUCTOS WHERE CODIGOPRODUCTO = :NEW.CODIGOPRODUCTO;

IF :NEW.UNIDADESPEDIDAS > stock THEN

RAISE\_APPLICATION\_ERROR(-20006, 'No hay suficiente stock para el producto.');

END IF;

END;

/